Math

UIUCDCS-R-77-888

UILU-ENG 77 1748

GENERATING BINARY TREES LEXICOGRAPHICALLY

by

S. Zaks

August 1977

DEPARTMENT OF COMPUTER SCIENCE
UNIVERSITY OF ILLINOIS AT URBANA-CHAMPAIGN · URBANA, ILLINOIS

GENERATING BINARY TREES LEXICOGRAPHICALLY*

by

S. Zaks

Department of Computer Science
University of Illinois at Urbana-Champaign
Urbana, Illinois 61801

August 1977

* This work was supported in part by the National Science Foundation under Grant Number MCS-73-03408.

TABLE OF CONTENTS

I. INTRODUCTION

We show a 1-1 correspondences among all the regular binary trees with n internal nodes, all the 0,1-sequences $x = \{x_i\}_1^{2n}$ of n 1's and n 0's in which the total number of 1's in each prefix $\{x_i\}_1^{\ell}$ ($\ell \leq n$) is at least as the total number of 0's, all the integer sequences $y = \{y_i\}_1^n$ in which $y_1 < y_2 < \ldots < y_n \leq 2n$ and $y_i \geq 2i$ for $i = 1,2,\ldots,n$, and all the integer sequences $z = \{z_i\}_1^n$ in which $0 < z_1 < z_2 < \ldots < z_n$ and $z_i \leq 2i - 1$ for $i = 1, 2, \ldots, n$.

The term "lexicographic ordering of trees" is discussed, and the relation between it and the lexicographic ordering of the corresponding sequences is shown.

For these sequences we (1) develop an algorithm which generates them lexicographically, (2) show how to find the position of a given sequence in the lexicographic ordering, and (3) show how to find a sequence given its position.

We prove (Theorem 4.8) that our order of generating these trees is equivalent to the order in which they are generated by two existing algorithms. The generating procedure is then generalized for the k-ary case.

Section 2 introduces basic definitions and notations, including two different definitions for "lexicographic ordering of trees." In Section 3 we discuss existing related algorithms. Section 4 establishes the 1-1 correspondences mentioned above, and algorithm which generates binary trees is presented. Sections 5 and 6 deal with the ranking function and the unranking procedure, respectively. In Section 7 we generalize the generating algorithm for k-ary trees. Additional comments and a summary of our discussions are presented in Section 8. A list of all the regular binary trees with 4 internal nodes and the associated corresponding structures and sequence are shown in the Appendix.

II. DEFINITIONS AND NOTATIONS

2.1 We state here defintions and notations that we shall use throughout this paper. We follow mainly the terminologies in [KN 1] and [LI 2].

2.2 While saying tree we mean an ordered tree. For $k \geq 2$ we define a k-ary tree T as follows: either T is empty or it has a distinguished node r called its root that is connected to $T_1, T_2, \ldots, T_k$, each of which is a k-ary tree. For $k = 2$ we write T_L and T_R instead of T_1 and T_2, respectively, which will be referred to as left subtree and right subtree. The root of each non-empty T_i for $1 \leq i \leq k$ is called a son of the root r, while r is its father. A vertex in a tree can be an internal node or an external node (also called a leaf). A regular k-ary tree is a non-empty k-ary tree with each internal node having k sons. We denote by $|T|$ the number of vertices of a tree T. We denote $T(k,n) = \{$all the regular k-ary trees with n internal nodes$\}$. $t(k,n)$ will be the number of elements in $T(k,n)$. We define $B_n = T(2,n)$ and $b_n = t(2,n)$.

2.3 There is a well-known 1-1 correspondence between B_n and all the binary trees with n vertices: From a tree $T \in B_n$ remove all the leaves and the edges incident with them. We get a binary tree with n vertices, and this transformation establishes the 1-1 correspondence. (See [KN 1: p.559] for this correspondence, and [TR 1: p.3] for an extension for k-ary trees).

2.4 We explain first what we mean by a "lexicographic order" of trees. The following definition is used in [KN 1: 2.3.1, ex. 25], [KNO: p. 113] and the first few sections of [TR 1]:

Definition 2.5: Given two k-ary trees T and T', we say that $T < T'$ if

(1) $|T| < |T'|$, or

(2) $|T| = |T'|$, and for some $1 \leq i \leq k$ we have

(a) $T_j = T_j'$ for $j = 1, 2, \ldots, i-1$, and

(b) $T_i < T_i'$

(See [TR 1: p. 16]). This defines a linear order in T(k,n). See Appendix for an example.

2.6 Unfortunately, no algorithm is known yet which generates those trees in order using this definition. In [KNO] and [TR 1] we have the ranking function and unranking procedure,which may be used in an indirect way to generate the trees lexicographically (given a tree, apply the ranking function to find its position, then apply the unranking procedure to build the tree that occupies the next position).

2.7 We define now another "lexicographic order" of trees. As will be shown later, all the existing algorithms which (directly) generate trees in "lexicographic order" are using this definition.

Definition 2.8: Given two k-ary trees T and T', we say that $T < T'$ if

(1) T is empty, or

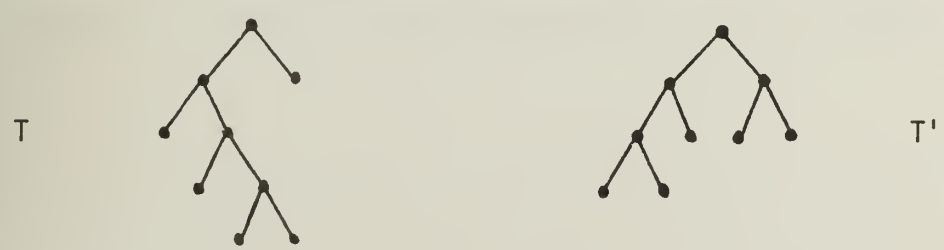
(2) T is not empty, and for some i, $1 \leq i \leq k$, we have

(a) $T_j = T'_j$ for $j = 1, 2, \ldots, i-1$, and

(b) $T_i < T'_i$.

This defines a linear order in T(k,n). See Appendix for an example.

2.9 As explained in 2.6, we use definition 2.8 throughout this work. The two definitions 2.5 and 2.8 are not equivalent. For example, $T <_{(2.8)} T'$ but $T' <_{(2.5)} T$ for the trees T and T' in Figure 1.



Ordering of Trees

Figure 1

2.10 We know that

$$t(k,n) = \frac{1}{(k-1)n+1}\binom{kn}{n}.$$

In particular,

$$b_n = \frac{1}{n+1}\binom{2n}{n}.$$

(See [KN 1: p. 584]).

2.11 Our discussion contains three parts:

(1) Generating step: we generate B_n lexicographically (according to the ordering as discussed above).

(2) Ranking step: we compute the function Index (T), which assigns to a given tree its appropriate position in this ordering.

(3) Unranking step: given a position x in this linear ordering, we construct the tree T s.t. Index (T) = x.

Step (1) is then generalized for an arbitrary k.

III. EXISTING ALGORITHMS

The following results discuss some or all the steps of 2.11:

3.1 In [RH] a binary tree is represented by the sequence of the level numbers of its leaves from left to right. Those sequences are then generated lexicographically (we show here - Theorem 4.8 - that the corresponding trees are generated thus in order according to definition 2.8.), and the ranking and unranking functions are discussed. This way of labeling can be extended to k-ary trees (see Theorem 7.8).

3.2 In [KN 1: 2.2.1, ex. 2,4,5; 2.3.1 ex. 6], [TR 1, 2] and [KNO] the numbers 1, 2, ... , 2n+1 are used to label the vertices of a tree $T \in B_n$ in some order (say, preorder: Root-Left-Right) and then these labels are read in another order (say, inorder: Left-Root-Right). We get a permutation of {1, 2, ... , 2n+1}. The preorder-inorder choice (used above in the brackets) is used in [TR 1,2], in which all the three steps of generating, ranking and unranking are discussed. Generalizations of the ranking and of the unranking steps for k-ary trees are discussed in [TR 1,2]. All those results use definition 1.8 for "lexicographic order" (see theorems 3.8 and 6.8). The various possible labelings are discussed in [KNO], which deals with ranking and unranking for binary trees, using definition 1.5. This definition is used - for ranking and unranking in the k-ary case - in [TR 1].

3.3 For the two labeling sequenses mentioned above, see example in the Appendix.

IV. TREES AND INTEGER SEQUENCES; THE GENERATING ALGORITHM

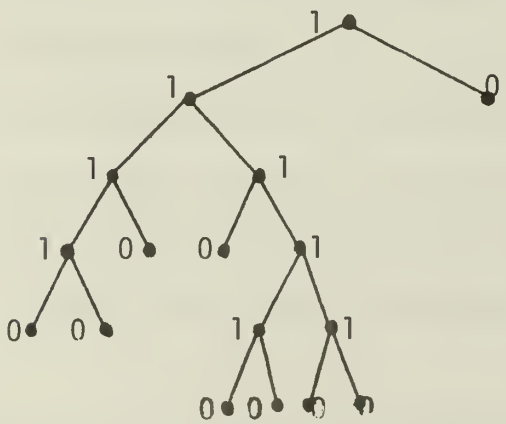
4.1 The number b_n of regular binary trees with n internal nodes is the n-th Catalan number, $b_n = \frac{1}{n+1}\binom{2n}{n}$. Given such a tree T, we label each internal node with 1 and each leaf with 0. We then read these labels in preorder (Root-Left-Right). We get a sequence of n 1's and n+1 0's. As the last visited node is always a leaf, we omit the corresponding 0. Denote the resulting sequence by $f(T) = \{x_i\}_1^{2n} = x$. From x we build a sequence $g(T) = \{y_i\}_1^n = y$ s.t.

y_i = position of the i-th 0 in x

and a sequence $h(T) = \{z_i\}_1^n = z$ s.t.

z_i = position of the i-th 1 in x.

When no confusion occurs, we omit commas and brackets in writing a sequence explicitly. See Figure 2 for an example.



x = 1111000101100100

y = 5, 6, 7, 9, 12, 13, 15, 16

z = 1, 2, 3, 4, 8, 10, 11, 14

Labeling a Binary Tree

Figure 2

A sequence $x = \{x_i\}_1^{2n}$ will be called feasible if there is a tree T s.t. x = f(T). The corresponding definition holds for the y and z sequences.

A 0,1-seuqnce is said to have the dominating property if in any prefix the number of 1's is at least as the number of 0's.

Theorem 4.2: A 0,1-sequence $x = \{x_i\}_1^{2n}$ is feasible iff

(1) It has n 1's and n 0's, and

(2) It has the dominating property.

Proof: If x is feasible then (1) holds clearly. For (2) we note that the number of internal nodes we visit, as described in 4.1, is always at least as the number of leaves (except in the last step, when we have visited n + 1 leaves but only n internal nodes; but this last leaf was excluded in x).

These two conditions are also sufficient, because the number of those sequences equal the number of those trees (see 4.4), and two different trees T and T' correspond to two different sequences x and x' by Theorem 4.8. □

4.3 It should be mentioned that associating the sequence f(T) with $T \in B_n$ can be found in other references (see [GA]; and [KN 3: p.63] for a related labeling). The correspondence between x = f(T) and z = h(T) is mentioned in [KN 3: p. 63], using the notion of Young's tableau. Our contribution is in making use of these sequences for the generating, ranking and unranking procedures and in generalizing these sequences and the generating procedure for k-ary trees.

4.4 The box-office problem* is usually related to those 0,1-sequences (see [YY: problem 83], [GN: p.36]), with the solution $\frac{1}{n+1}\binom{2n}{n}$. Other interpretations of these sequences, such as Bertrand's balloting problem, are known (see [LI 1: p. 74], [GA]).

4.5 The following theorem establishes the relation between binary trees and integer sequences:

Theorem 4.6: The following sets are in a 1-1 correspondence with one another:

(1) B_n

---

* The box-office problem: 2n people are waiting in a line at a box office. n of them have 1 dollar bills, and the rest have 2 dollar bills. Tickets cost 1 dollar each. When the box-office opens, there is no money in the till. Each customer buys one ticket. In how many ways can they stand such that none of them will have to wait for change?

(2) All the 0,1-sequences $x = \{x_i\}_1^{2n}$, with n 1's and n 0's, having the dominating property.

(3) All the integer sequences $y = \{y_i\}_1^n$, s.t. $y_1 < y_2 < \ldots < y_n \leq 2n$ and $y_i \geq 2i$ for $i = 1, 2, \ldots, n$.

(4) All the interger sequences $z = \{z_i\}_1^n$, s.t. $0 < z_1 < z_2 < \ldots < z_n$ and $z_i \leq 2i - 1$ for $i = 1, 2, \ldots, n$.

Proof: (1) ≡ (2)

See Theorem 4.2.

(2) ≡ (3)

With a 0,1-sequence $x = \{x_i\}_1^{2n}$, we associate a corresponding sequence $\{y_i\}_1^n$ as explained in 4.1.

Suppose x satisfies (2). If for some ℓ, $1 \leq \ell \leq n$, $y_\ell < 2\ell$, then among $x_1, x_2, \ldots, x_{2\ell-1}$ there are at least ℓ 0's, which violates the dominating property of x. Hence $y_i \geq 2i$ for $i = 1, 2, \ldots, n$.

Conversely, assume x doesn't satisfy (2). It it doesn't have n 0's its corresponding y(x) doesn't satisfy (3). If it has n 1's and n 0's but doesn't have the dominating property, then for some ℓ, $1 \leq \ell \leq 2n$, the number t of 0's in a prefix $\{x_i\}_1^\ell$ is more than the number $\ell - t$ of 1's, or $\ell < 2t$. Hence the position y_t of the t-th 0 staisfies $y_t \leq \ell < 2t$, hence y doesn't satisfy (3).

It is clear that different sequences x and x' correspond to different sequences y and y', respectively, and this completes the proof.

(3) ≡ (4)

Left to the reader. □

4.7 We define the lexicographic order for sequences as usual: Let $u = \{u_i\}_1^n$ and $v = \{v_i\}_1^n$. We say that $u < v$ if there exist an index ℓ, $1 \leq \ell \leq n$, s.t. $u_j = v_j$ for $j = 1, 2, \ldots, \ell-1$ and $u_\ell < v_\ell$.

The following observation is essential for our work:

Theorem 4.8: Let T and T' be two regular binary trees with n internal nodes, and let the corresponding sequences x, y, z and x', y', z' be as defined above. The following are equivalent:

(1) $T < T'$ (definition 2.8)

(2) $x < x'$

(3) $y < y'$

(4) $z > z'$

(5) $T < T'$ according to Ruskey-Hu's level sequence (see 3.1)

(6) $T < T'$ according to Trojanowski's permutation (see 3.2)

Proof: (2) $\Leftrightarrow$ (3) $\Leftrightarrow$ (4)

Immediately from the definition of y and z (4.1).

(1) $\Leftrightarrow$ (2)

$T < T'$ iff the ℓ^{th} vertex visited is a leaf in T but is an internal node in T', while the first $\ell - 1$ vertices are the same, for some ℓ, $1 < \ell < 2n$. This happens iff $x_i = x'_i$ for $i = 1, 2, \ldots, \ell-1$ and $x_\ell = 0 < 1 = x'_\ell$ for some ℓ as above, which happens iff $x < x'$.

(2) $\Leftrightarrow$ (5)

$x < x'$ iff for some ℓ, $1<\ell<2n$, $x_i = x'_i$ for $i = 1,2, \ldots, \ell-1$, and $x_\ell = 0<1 = x'_\ell$. Let us look at those parts of T and T', corresponding to the first $\ell - 1$ nodes traversed in preorder. Those parts must be isomorphic, otherwise we could not have $(x_1, \ldots, x_{\ell-1})=(x'_1,\ldots,x'_{\ell-1})$, by the equivalence (1)$\Leftrightarrow$(2). Hence, $x<x'$ iff all the leaves among the first $\ell - 1$ nodes (in preorder traversing) of T and T' are in the same levels, correspondingly, but the leaf now scanned in T (corresponding to $x_\ell = 0$) is in a lower level than that of the leaf that will be next scanned in T' (this leaf will be a successor of the internal node corresponding to $x'_\ell = 1$); this happens iff the level sequence of T is less than that of T' in Ruskey-Hu's ordering.

(2) $\Leftrightarrow$ (6)

$x < x'$ iff for some ℓ $x_i = x'_i$ for $i = 1, 2, \ldots, \ell-1$ and $x_\ell = 0 < 1 = x'_\ell$. This happens iff while scanning both T and T' in preorder we meet internal nodes and leaves correspondingly, but the ℓ-th scanned node is a leaf b in T and an internal node b' in T'. This happens iff while reading the preorder labeling of the vertices of T and T' in inorder , the two corresponding permutations p and p' coincide until we read the labels of b and b'. At this point in p we add the label of b as the next element, while in p' we add a number greater than the label of b', but the labels of b and b' are the same. This happens iff $p \prec p'$. □

Corollary 4.9: In order to generate B_n (lexicographically), it is sufficient to generate all the above sequences x, y (lexicographically) or z(antilexicographically) of Theorem 4.6.

Note: It should be clear how the conversion of a sequence into its corresponding binary tree is made, and this fact is not discussed here.

4.10: It is interesting to compare the feasibility condition in [RH] with the following:

Theorem 4.11: For a 0,1-sequence $x = \{x_i\}_1^{2n}$ the following are equivalent:

(1) x is a feasible sequence.

(2) Replacing the left most 100 pattern in x by 0, as long as possible, terminates in $(10)^t$ for some $1 \leq t \leq n$.

(3) Erasing any 10 pattern in x, as long as possible, terminates in the empty sequence.

Proof: Left to the reader. □

4.12: The following algorithm generates the sequences lexicographically:

Algorithm 4.13: (Generating the z sequences lexicographically):

Step 1: Begin with $z = \{z_i\} = \{1, 2, \ldots, n\}$

Step 2: Scan the sequence from right to left.

Find the right most index j s.t. $z_j < 2_j - 1$. If no such index exists - go to step (5).

Step 3: The sequence $z' = \{z'_i\}$ next to z is built as follows:

$$z'_i \leftarrow z_i \text{ for } i < j$$

$$z'_j \leftarrow z_j + 1$$

$$z'_i \leftarrow z_{i-1} + 1 \quad \text{for } i = j + 1, \ldots, n \ .$$

Let z' be called z.

Step 4: Go to step (2).

Step 5: Exit.

Theorem 4.14: Algorithm 4.13 generates all the feasible sequence $z = \{z_i\}_1^n$ s.t. $0 < z_1 < z_2 < \ldots < z_n$ and $z_i \leq 2i - 1$ for $i = 1, 2, \ldots, n$.

Proof: Trivial; the generating procedure in the algorithm follows the definition of lexicographic order. □

See Appendix for an example.

V. THE RANKING FUNCTION

5.1 In this section we show how to determine the position Index(T) of a given tree $T \in B_n$ in the lexicographic ordering of B_n. For this purpose, we convert the tree to its correspoinding z seqence as discussed in Section 4, find the poisition index(z) of z in the lexicographic ordering of all these sequences, and set Index (T) = b_n - index(z) + 1.

5.2 To determine the ranking function we define the doubly indexed sequence $a_{i,j}$, $0 \leq j \leq i-1$:

$$a_{i,i-1} = 1 \qquad \text{for all } i ,$$

$$a_{i,0} = b_i = \frac{1}{i+1}\binom{2i}{i} \qquad \text{for all } i ,$$

$$a_{i,j} = a_{i,j+1} + a_{i-1,\,j-1} \qquad \text{otherwise} .$$

i \ j	0	1	2	3	4	5	6	7
1	1							
2	2	1						
3	5	3	1					
4	14	9	4	1				
5	42	28	14	5	1			
6	132	90	48	20	6	1		
7	429	297	165	75	27	7	1	
8	1430	1001	572	275	110	35	8	1

The Sequence $a_{i,j}$

Figure 3

5.3 Given a sequence $z = \{z_i\}_1^n$, let $\ell = \text{init}(z)$ denote the largest i s.t. $z_i = i$ (note that $z_1 = 1$, so $\text{init}(z) \geq 1$). Let $\bar{z} = \{\bar{z}_i\}_1^{n-1}$ be the sequence built from z by deleting z_ℓ and setting $\bar{z}_j \leftarrow z_j$ for $j < \ell$, $\bar{z}_j \leftarrow z_{j+1} - 2$ for $j > \ell$. The following theorem is the key to the ranking algorithm:

Theorem 5.4: The following definition of the function index(z) assigns to a feasible sequence $z = \{z_i\}_1^n$ its position in the lexicographic ordering of all these sequences:

$$\text{index}(z) = \begin{cases} 1 & \text{if } \ell = n \\ a_{n,\ell} + \text{index}(\bar{z}) & \text{if } \ell \neq n \end{cases}$$

Proof: It is clear that

$$\begin{aligned} \text{index}(z) = &[\text{number of feasible sequences } \tilde{z} \\ &\text{s.t. } \text{init}(\tilde{z}) \geq \ell + 1] + \\ &[\text{position of } z \text{ among all those feasible} \\ &\text{sequences } \tilde{z} \text{ for which } \text{init}(\tilde{z}) = \ell]. \end{aligned}$$

We show that the first term in this sum equals $a_{n,\ell}$ and that the second one is $\text{index}(\bar{z})$ (it is clear that if $\ell = n$ then $\text{index}(z) = 1$ - see step 1 in Algorithm 4.13).

First term For $\ell = 0$: $a_{n,0} = \frac{1}{n+1}\binom{2n}{n}$ is the number of sequences that begin with 1, which is all our sequences. For $\ell = n-1$: $a_{n,n-1} = 1$ and there is really only one sequence $\tilde{z}$ s.t. $\text{init}(\tilde{z}) \geq \ell + 1 = n$, namely $\{1, 2, \ldots, n\}$.

For any other ℓ the sequences $\tilde{z}$ with $\text{init}(\tilde{z}) \geq \ell + 1$ are partitioned into those for which $\text{init}(\tilde{z}) > \ell + 1$ and those for which $\text{init}(\tilde{z}) = \ell + 1$, and this is taken case by the relation $a_{n,\ell} = a_{n,\ell+1} + a_{n-1,\ell-1}$. We leave the rest of the induction details to the reader.

Second term All the sequences $\tilde{z}$ s.t. $\text{init}(\tilde{z}) = \ell$ corresponds to a 0,1-sequence $x = \underbrace{11 \ldots 10}_{\ell \quad 2n-\ell} \ldots$. The sequence $\bar{z}$ as described above is exactly the

sequence that we get from x by omitting the first 10 patterns from the left (encircled above). So the position of z among all the sequences $\tilde{z}$ with $init(\tilde{z}) = \ell$ is exactly $index(\overline{z})$. □

Example 5.5: Given the tree in Figure 2, we have:

$$index(1,2,3,4,8,10,11,14) = a_{8,4} + index(1,2,3,6,8,9,12)$$
$$index(1,2,3,6,8,9,12) = a_{7,3} + index(1,2,4,6,7,10)$$
$$index(1,2,4,6,7,10) = a_{6,2} + index(1,2,4,5,8)$$
$$index(1,2,4,5,8) = a_{5,2} + index(1,2,3,6)$$
$$index(1,2,3,6) = a_{4,3} + index(1,2,4)$$
$$index(1,2,4) = a_{3,2} + index(1,2)$$
$$index(1,2) = 1$$

so

$$index(1,2,3,4,8,10,11,14) = a_{8,4} + a_{7,3} + a_{6,2} + a_{5,2} + a_{4,3} + a_{3,2} + 1 = 110 + 75 + 48 + 14 + 1 + 1 + 1 = 250.$$

Thus $Index(T) = b_8 - 250 + 1 = 1181$.

5.6 The solution to the recurrence relation for $a_{i,j}$ is

$$a_{i,j} = \frac{j+2}{2i-j}\binom{2i-j}{i-j-1}$$

(See [RH] for an analytic approach, or apply the technique in [YY: problem 83] for a combinatorial approach , which is also found in [WH]) .

VI. THE UNRANKING PROCEDURE

6.1 Given a number i, $1 \leq i \leq b_n$, we show in this section how to find the regular binary tree T with n internal vertices s.t. Index(T) = i.

As explained in 5.1, it is enough if we know how to do it for a z sequence.

6.2 The algorithm we are presenting follows immediately from the way we calculated index(z) using Theorem 5.4, so we shall skip the proof of it here. We first bring an example:

Example 6.3: To find $T \in B_8$ s.t. Index(T) = 1181, we note that it is sufficient to find the appropriate z sequence for which index(Z) = b_8 - Index(T) + 1 = 250. By Theorem 5.4 we know that

$$250 = a_{8,\ell} + \text{index}(z)$$

As a consequence from Section 5, we choose ℓ s.t.

$$a_{8,\ell} < 250 \leq a_{8,\ell-1}$$

Here we choose $\ell = 4$, and get $250 = 110 + \text{index}(\bar{z})$, or $\text{index}(\bar{z}) = 90$, where $\bar{z}$ is a sequence corresponding to a tree $\bar{T} \in B_7$. Next we get

$90 = a_{7,3} + \text{index}(\bar{\bar{z}})$, or $\text{index}(\bar{\bar{z}}) = 15$, and so on.

At the end we have

$$250 = a_{8,4} + a_{7,3} + a_{6,2} + a_{5,2} + a_{4,3} + a_{3,2} + a_{2,1}.$$

From this decomposition of 250, we reconstruct the sequence z:

The last 1 tells us that at the very end (of computing index(z)) we had {1,2}. Then $a_{3,2}$ tells us that a step before we had the sequence {1,2,t}, and that after omitting the 2 and setting $t \leftarrow t - 2$ we get {1,2}. So we have {1,2,4}. Recurring in this manner, we get:

$$a_{4,3} \rightarrow 1,2,3,6$$

$$a_{5,2} \rightarrow 1,2,4,5,8$$

$$a_{6,2} \to 1,2,4,6,7,10$$

$$a_{7,3} \to 1,2,3,6,8,9,12$$

$$a_{8,4} \to z = \{1,2,3,4,8,10,11,14\}$$

and now we build the tree (see Figure 2, and compare this example with Example 5.5).

6.4 We bring here the algorithm that converts a number to its corresponding z sequence. We are given n and t, and look for a sequence $z = \{z_i\}_1^n$ ($0 < z_1 < z_2 < \ldots < z_n$, $z_i \leq 2i - 1$ for $i = 1, 2, \ldots, n$) s.t. index(z) = t :

Algorithm 6.5:

Step 1: $A \leftarrow t$, $j \leftarrow n$

Step 2: Find ℓ_j s.t. $a_{j,\ell_j} < A \leq a_{j,\ell_j - 1}$

Step 3: $A \leftarrow t - a_{j,\ell_j}$

$j \leftarrow j - 1$

if $A > 0$ then go to Step 2.

Step 4: We have now $t = \sum_{j=j_0}^{n} a_{j,\ell_j}$

Step 5: Set $z \leftarrow \{1,2, \ldots, j_0\}$

$m \leftarrow j_0 + 1$, $s \leftarrow \ell_m$

Step 6: Change z as follows:

$z_i \leftarrow z_i$ $\qquad i < s$

$z_{i+1} \leftarrow z_i + 2$ $\qquad i = m, m-1, \ldots, s$

$z_s \leftarrow s$

Step 7: $m \leftarrow m + 1$

if $m \leq n$ then $s \leftarrow \ell_m$ and go to step 6.

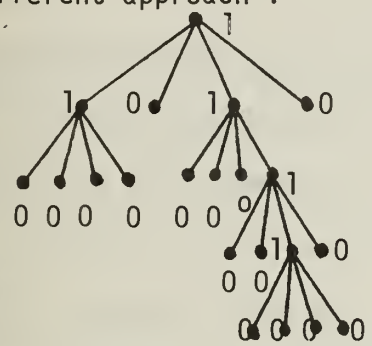
Step 8: Exit.

VII. K-ARY TREES

7.1 We generalize in this section the results about correspondence between trees and integer sequences and the generating procedure, as discussed in Section 4, to k-ary trees. The proofs in this section are omitted.

7.2 Given a tree $T \in T(k,n)$ we label its nodes as in the binary case, and get a 0,1-sequence $f(T) = \{x_i\}_1^{kn} = x$ and an integer sequence $h(x) = \{z_i\}_1^{n} = z$ (the sequence y is not discussed here). See Figure 5 for an example.

A sequence related to f(T) is associated with a planted planar tree in [KL], using a different approach .



x = 11000001000100100000

z = 1,2,8,12,15

Labeling a k-ary Tree

Figure 4

A 0,1-sequence so obtained will be called a k-feasible sequence. It will be characterized as having the following property, as is shown in Theorem 7.6.

7.3 Let $a = \{a_i\}_1^{kn}$ be a seqence consisting of n 1's and (k-1)n 0's. We say that a has the k-dominating property if in each initial subsequence $\{a_i\}_1^{\ell}$, $1 \leq \ell \leq kn$, the accumulated number of 1's is at least $\lceil \frac{\ell}{k} \rceil$ ($\lceil t \rceil$ means "the smallest integer greater or equal to t"). In other words, if this subsequence contains s 1's, then it contains at most (k-1)s 0's. Note that for k = 2, we get the dominating property as defined in 4.1.

7.4 It will be of interest to mention the following generalization of the

box-office problem: The generalized box-office problem*: kn people are waiting in line at a box-office. n of them have k-1 1 dollar bills and the rest (k-1)n have k dollar bills (assuming that k dollar bills do exist). Each ticket costs k - 1 dollars. When the box-office opens, there is no money in the till. Each customer buys one ticket. In how many ways can they stand such that none of them will have to wait for change ?

Clearly, for k = 2 we get the previous box-office problem of 4.4. Furthermore as a result of theorem 7.8, the answer to the generalized box-office problem is $\frac{1}{(k-1)n+1}\binom{kn}{n}$.

7.5 The following theorem makes the desired connection between k-ary trees and the corresponding integer sequences and is a generalization of theorem 4.6:

Theorem 7.6: The following sets are in -1 correspondence with one another:

(1) All the regular k-ary trees with n internal vertices.

(2) All the 0,1-sequences with n 1's and (k-1)n 0's $\{x_i\}_1^{kn}$, having the k-dominating property.

(3) All the integer sequences $\{z_i\}_1^n$ s.t. $0 < z_1 < z_2 < \ldots < z_n$ and $z_i \leq ki - (k-1)$ for i = 1, 2, ... , n.

7.7 The follwoing generalization of Theorem 4.8 is the basis for our generating discussion. The proof follows immediately from that of Theorem 4.8. Note that in [TR1, 2] the permutation associated with a tree is not immediately extended from the binary case. Still, the order is the same.

Theorem 7.8: Let T and T' be two regular k-ary trees with n internal nodes, and let the corresponding x, z and x', z' sequences be as defined above. The following are equivalent:

(1) T < T' (definition 2.8)

(2) x < x'

(3) z > z'

* See [DM] for a related problem .

(4) $T < T'$ according to Ruskey-Hu's level sequence as extended to k-ary trees (see 3.1).

(5) $T < T'$ according to Trojanowski's permutation as extended to k-ary tree (see 3.2).

Corollary 7.9: In order to generate (lexicographically) all the regular k-ary trees with n internal nodes, it is sufficient to generate all the sequences x(lexicographically) or z(antilexicographically) of Theorem 7.6.

The note after corollary 4.9, and the discussion in 4.10 - 4.11 (replacing 100 and 10 patterns with $1\underbrace{00 \ldots 0}_{k}$ and $1\underbrace{00 \ldots 0}_{k-1}$ patterns respectively), should be applied here as well.

Algorithm 7.10: (Generating the z sequences lexicographically): Exactly as algorithm 4.13, with one change. In step (2) replace the upper bound $2j - 1$ by $kj - (k-1)$.

7.11 All the discussion in Section 4, following algorithm 4.13, applies here.

7.12: As for the ranking function and unranking procedure we were not able to extend the algorithms from the binary case to the k-ary case; yet, as follows from Theorem 7.8, we can use the appropriate algorithms in [TR 1, 2] for these tasks. It should be noted that generating the k-ary trees by the z sequences, is simpler than that of level numbers ([RH]) or permutations ([KN 1], [TR 1, 2], [KNO]).

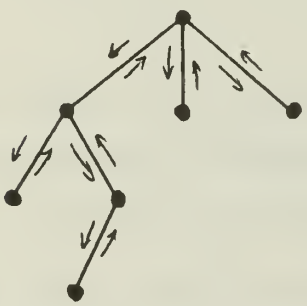
7.13: Two methods for ranking and unranking k-ary trees - one of which is presented among other combinatorial objects ([ZR]), and the other performs these tasks in an order different from the two definitions, as discussed in Section 2 ([ZA]) - are a subject of future work.

VIII. SUMMARY

8.1 After discussing the term "lexicographic order of trees," we have seen algorithms that generate all the regular binary trees with n internal vertices lexicographic order, find the position of a given tree in this ordering and find a tree given its position. For regular k-ary trees we treated the generating algorithm.

8.2 As explained in 2.2, there is a simple 1 - 1 correspondence between all the regular k-ary trees with n internal vertices and all the k-ary trees with n vertices, so the results mentioned in 8.1 can be extended to this new class.

8.3 If we take any tree with n edges and travese it in the way shown in Figure 5, (see [BN]),when 1 means "going down" and 0 means "going up", we get a 0,1-sequence



$\Rightarrow$ 110110001010

Figure 5

which have the dominating property - the number of "going down" must be at least as that of "going up," otherwise we will get above the root! Consequently, our algorithms can be used to generate, rank and unrank all the trees with n edges. See Appendix for an example.

8.4 In finding the ranking function we defined the sequence $a_{i,j}$ and then proved that $a_{n,\ell}$ is the number of feasible sequences z s.t.init(z) $\geq \ell + 1$. We could as well choose the alternative way: define a_{ij} by this last property, and then prove that $a_{i,j}$ satisfies the relations by which we defined it.

8.5 The following remark concerns the difference between the two ordering of trees (definitions 2.5 and 2.8): Given 2 trees T and T', to determine whether $T < T'$, we scan both trees in preorder. Using definition 2.5, we compare the sizes of the subtrees rooted at the concurrent nodes; in other words, we use a global information concerning those nodes. On the other hand, using definition 2.8 we compare the characters of the concurrent nodes (whether they are internal nodes or leaves); in other words, we use local information concerning those nodes.

ACKNOWLEDGE

I wish to thank Professor C. L. Liu for many helpful suggestions while working on this paper.

APPENDIX

We show here all the 14 regular binary trees with 4 internal nodes in lexicographic order. To each of them we assign the structures and sequences introduced in the paper, as follows:

(1) The x sequence (0, 1-sequence)

(2) The y sequence

(3) The z sequence

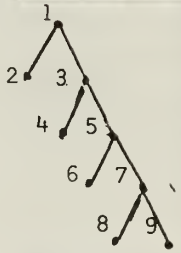
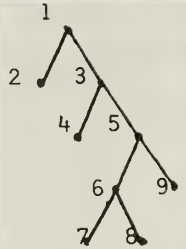
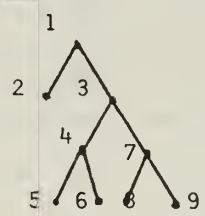
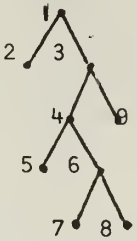
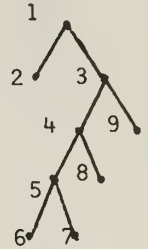
(4) Ruskey and Hu's level sequence

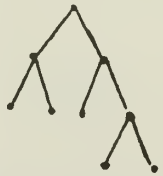
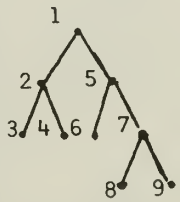
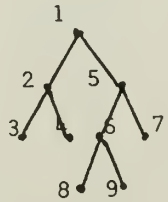
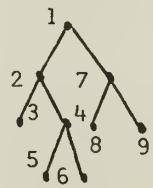
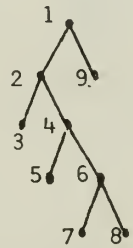
(5) Trojanowski's permutation
(note that reading only the internal vertices in Trojanowski's labeling in preorder (Root-Left-Right) gives us again the corresponding z sequence, and the reasoning for this is left to the reader.)

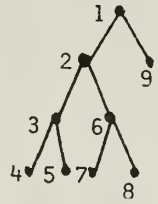
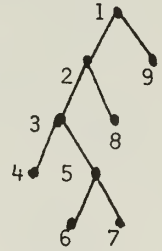
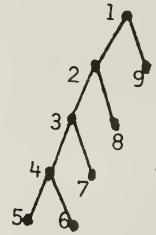
(6) The corresponding binary tree with 4 vertices.

(7) The corresponding tree with 4 edges.

(8) The index while being ordered according to definition 2.5.

Definition 2.8)	1	2	3	4	5
Regular Binary Trees with 4 internal Vertices					
Binary Trees with 4 Vertices					
x-sequence	10101010	10101100	10110010	10110100	10111000
y-sequence z-sequence	1357 2468	1356 2478	1347 2568	1346 2578	1345 2678
Ruskey & Hu Level Sequence	12344	12443	13333	13442	14432
Preorder Labeling Inorder Traversing (Knuth, Trojanowski, and Knot)	 214365879	 214376859	 215463879	 215476839	 216574839
Trees with 4 Edges					
Index (According to Definition 2.5)	1	2	3	4	5

Index (According Definition 2.8)	6	7	8	9	10
Regular Binary Trees with 4 Internal Vertices					
Binary Trees with 4 Vertices					
x-seqence	11001010	11001100	11010010	11010100	11011000
y-sequenece z-sequence	1257 3468	1256 3478	1247 3568	1246 3578	1245 3678
Ruskey & Hu Level Sequence	22233	22332	23322	23441	24431
Preorder Labeling Inorder Traversing (Knuth, Trojanowski, and Knot)	 324165879	 324176859	 325461879	 325476819	 326574819
Trees with 4 Edges					
Index (According to Definition 2.5)	6	7	8	10	11

Index (According to Definition 2.8)	11	12	13	14
Regular Binary Trees with 4 Internal Vertices				
Binary Trees with 4 Vertices				
x-sequence	11100010	11100100	11101000	11110000
y-sequence z-sequence	1237 4568	1236 4578	1235 4678	1234 5678
Ruskey & Hu Level Sequence	33222	33331	34421	44321
Preorder Labeling Inorder Traversing (Knuth, Trojanowski, and Knot)	 435261879	 435276819	 436572819	 54637289
Trees with 4 Edges				
Index (According to Definition 1.5)	9	12	13	14

REFERENCES

[BM] N. G. DeBruijn and B. J. M. Morselt, "A Note on Plane Trees," Journal of Combinaotiral Theory 2, 27-34 (1967).

[DM] A. Dvoretzky and T. Motzkin, "A Problem of Arrangements," Duke Math. Journal 14 (1947), 305-313.

[GA] M. Gardner, "Mathematical Games": Catalan Numbers, Scientific American, June 1976, pp. 120-122.

[GN] B. N. Gnedenko, The Theory of Probability, Chelsea Publishing Company, N.Y., N.Y., 1962.

[KL] D. A. Klarner, "Correspondences Between Plane Trees and Binary Sequences," Journal of Combinatorial Theory 9, 401-411 (1970)

[KNO] Gary D. Knott, "A Numbering System for Binary Trees," Communications of the ACM, Vol. 20, Number 2, February 1977.

[KN1] Donald E. Knuth, The Art of Computer Programming Vol. 1: Fundamental Algorithms, Addison-Wesley, Reading, MA 1968.

[KN3] Donald E. Knuth, The Art of Computer Programming Vol. 3: Sorting and Searching, Addison-Wesley, Reading, MA 1973.

[LI1] C. L. Liu, Topics in Combinatorial Mathematics, Mathematical Association of America, 1972.

[LI2] C. L. Liu, Elements of Discrete Mathematics, McGraw-Hill, 1977.

[RH] F. Ruskey and T. C. Hu, "Generating Binary Trees Lexicographically," SIAM Journal on Computing, to appear.

[TR1] Anthony E. Trojanowski, "On the Ordering, Enumeration and Ranking of k-ary Trees," Tech. Report UIUCDCS-R-77-850, Department of Computer Science, University of Illinois at Urbana-Champaign, February, 1977.

[TR2] Anthony E. Trojanowski, "Ranking and Listing Algorithms for k-ary Trees," SIAM Journal on Computing, to appear.

[WH] W. A. Whitworth, "Arrangements of m Things of One Sort and m Things of Another Sort Under Certain Conditions of Priority," Messenger of Math. 8 (1878), 105-114.

[YY] A. M. Yaglom and I. M. Yaglom, Challenging Mathematical Problems with Elementary Solutions Vol. 1: Combinatorial Analysis and Probability Theory, Holden-Day, 1964.

[ZA] S. Zaks, "Generating k-ary Trees Lexicographically," to appear.

[ZR] S. Zaks and D. Richards, "Generating Trees and Other Combinatorial Objects Lexicographically," to appear.

BIBLIOGRAPHIC DATA SHEET	1. Report No. UIUCDCS-R-77-888	2.	3. Recipient's Accession No.
Title and Subtitle			5. Report Date October, 1977
			6.
Author(s) S. Zaks			8. Performing Organization Rept. No.
Performing Organization Name and Address Department of Computer Science University of Illinois at Urbana-Champaign Urbana, IL 61801			10. Project/Task/Work Unit No.
			11. Contract/Grant No. MCS-73-03408
Sponsoring Organization Name and Address National Science Foundation Washington, DC			13. Type of Report & Period Covered
			14.

Supplementary Notes

Abstracts

We show a one-one correspondence between all the regular binary trees with n internal nodes and certain integer sequences, an algorithm for generating these trees lexicographically, and the ranking function and the corresponding unranking procedure. We then extend the generating algorithm to k-ary trees. Relations to existing algorithms are discussed.

Key Words and Document Analysis. 17a. Descriptors

k-ary tree, ordered tree, lexicographic order, ranking, and unranking

Identifiers/Open-Ended Terms

COSATI Field/Group

Availability Statement	19. Security Class (This Report) UNCLASSIFIED	21. No. of Pages
	20. Security Class (This Page UNCLASSIFIED	22. Price